Parallélisation d'un algorithme de détection de mouvement sur une architecture MIMD

Fabrice Heitz, Etienne Mémin, Thierry Priol, Sergui Jufresa

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

1992 25ème anniversaire

N° 1738

Programme 4
Robotique, Image et Vision

PARALLÉLISATION D'UN ALGORITHME DE DÉTECTION DE MOUVEMENT SUR UNE ARCHITECTURE MIMD

Fabrice HEITZ
Sergui JUFRESA
Etienne MEMIN
Thierry PRIOL

Juillet 1992

RR-1738

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX FRANCE
Tél. : 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Parallélisation d'un algorithme de détection de mouvement sur une architecture MIMD

Parallel Implementation of a Motion Detection Algorithm on a MIMD Architecture

Fabrice HEITZ, Sergui JUFRESA, Etienne MEMIN, Thierry PRIOL

IRISA/INRIA
Campus de Beaulieu
35042 Rennes Cedex

Publication Interne n° 669 - Juillet 1992 - 34 pages
Programme 4

Résumé

L'analyse d'images par modèles markoviens conduit généralement à la mise en oeuvre d'algorithmes de relaxation (déterministe ou stochastiques). L'algorithme *HCF*, s'appuyant sur la gestion d'une pile d'instabilité des sites de l'image, s'est révélé efficace pour accélérer les traitements. Dans ce rapport nous étudions la parallélisation de cette classe d'algorithme, dans le cadre d'une application de détection du mouvement. Il est montré que ce type d'algorithme se prête bien à une implémentation parallèle sur une architecture MIMD. Les résultats d'une expérimentation sur un hypercube iPSC/2 sont analysés.

Abstract

Markov Random Field-based image analysis generally leads to the implementation of relaxation algorithms. In this paper, the parallelization of the standard *HCF* (Highest Confidence First) relaxation algorithm is considered. This algorithm makes use of a stability heap in which image sites are ordered according to their energy level. It is shown that this algorithm is well-suited for MIMD architectures. The parallelization scheme is illustrated on a motion detection algorithm involving spatio-temporal processing of an image sequence. Experimental results on a hypercube iPSC/2 are provided.

1 Introduction

La théorie des champs de Markov associée à des approches bayesiennes globales de l'estimation statistique a permis de définir un cadre mathématique cohérent et efficace dans nombre de problèmes d'analyse d'image tels que la restauration d'image, la stéréovision, l'analyse du mouvement... L'intérêt des champs markoviens réside à la fois dans le caractère global, non-linéaire et non-causal de la modélisation, et dans le caractère local des calculs que leur mise

en œuvre entraîne. Ces modèles nécessitent l'emploi de méthodes de relaxation stochastiques ou déterministes qui visent à minimiser une fonction d'énergie globale dépendant d'un grand nombre de variables. Ces algorithmes conduisent à des temps de calcul qui peuvent être importants sur les machines séquentielles classiques. Leurs caractéristiques laissent cependant espérer des implémentations parallèles particulièrement efficaces.

Dans ce rapport, la mise en œuvre d'une version parallèle d'un algorithme de détection du mouvement reposant sur une modélisation markovienne est décrite. Dans le paragraphe 2, nous présentons le cadre de la modélisation markovienne ainsi que la version séquentielle de l'algorithme étudié. L'algorithme utilise une technique de relaxation déterministe, basée sur la gestion d'une pile d'instabilité des sites de l'image pour accélérer les traitements. Ce type d'algorithme se prête tout particulièrement à une implémentation parallèle sur une architecture MIMD. Le paragraphe 3 décrit plusieurs stratégies pour une mise en œuvre sur une architecture parallèle à mémoire distribuée. Enfin le paragraphe 4 présente les résultats d'une expérimentation réalisée sur un hypercube iPSC/2.

2 Détection du mouvement: une approche markovienne

2.1 La détection du mouvement

Le rôle dévolu à un système de vision dans de très nombreux contextes d'application (en particulier en robotique mobile, télésurveillance, contrôle du trafic routier, etc.) est de repérer, de suivre, voire d'identifier des objets en mouvement ou plus généralement des phénomènes temporels. Les problèmes relatifs à la surveillance de scènes dynamiques relèvent de l'analyse du mouvement apparent dans une séquence d'images, [1, 4].

L'analyse du mouvement apparent se structure hiérarchiquement en trois sous-problèmes fondamentaux: détection du mouvement, [5], estimation des déplacements apparents, [9] et segmentation au sens du mouvement, [7].

Lorsque la caméra est fixe, mouvement dans la scène et mouvement dans l'image sont étroitement liés. Il devient dans ce cas particulièrement intéressant de s'attaquer au problème de la détection du mouvement apparent, [5]. En effet, si l'on considère le problème générique de l'analyse de scène dynamique comme le repérage d'objets mobiles dans la scène, il est possible d'accéder directement à la localisation de ces éléments mobiles.

La détection du mouvement apparent dans une séquence d'images est évidemment un problème binaire. Il s'agit de distinguer les zones en mouvement des zones fixes. Il apparaît immédiatement que les changements temporels de la fonction intensité joueront un rôle important dans tout procédé de détection du mouvement apparent. S'il y a mouvement, interviennent généralement des variations temporelles de l'intensité. La réciproque n'est toutefois pas vraie. On ne peut pas assimiler carte binaire des changements temporels de l'intensité et carte binaire des masques des entités en mouvement dans l'image. En effet, d'une part, des modifications de l'illumination, voire du "bruit", peuvent entraîner de telles variations tem-

porelles. D'autre part, le mouvement d'un objet donne naissance à trois types de zones dans une carte de changements temporels : 1) le fond découvert par l'objet ; 2) le fond recouvert par l'objet ; 3) la zone de glissement de l'objet sur lui-même (les régions de ce dernier type étant le plus souvent très imparfaitement détectées, surtout lorsque la surface de l'objet est relativement uniforme). La détection du mouvement peut donc se diviser en deux sous-problèmes : la détection des changements temporels puis la reconstruction complète des projections des objets mobiles. Dans l'algorithme considéré ici, [5], ces deux sous-problèmes sont résolus de façon simultanée en utilisant des techniques de modélisation statistique associées à la théorie de l'estimation bayesienne.

2.2 Modélisation markovienne et estimation bayesienne

Le formalisme des champs markoviens permet l'intégration de sources d'information multiples et offre un traitement unifié de nombreux problèmes d'analyse d'images. En particulier, il autorise la spécification d'interactions *non-linéaires* entre primitives-image de natures *différentes*. Ces interactions sont généralement définies sur un voisinage réduit ν autour d'un point ce qui leur confère un caractère local. Si la spécification de la modélisation est locale (ce qui permet de réduire la charge de calcul), les modèles induits sont par contre globaux et peuvent être considérés comme généraux.

Dans notre cas, ces potentiels traduiront l'homogénéité spatiale des régions de mouvement détectées, ainsi que leurs propriétés temporelles. L'utilisation des champs markoviens se rattache, en cela, à la théorie de la régularisation, qui vise à introduire des contraintes *a priori* sur les solutions d'un problème sous-contraint ou mal conditionné.

Si $o = (o_1, o_2, o_M)$ désigne le champ des observations à partir duquel on cherche à identifier un champ d'étiquettes (non observé) $e = (e_1, e_2, e_N)$, la théorie des champs markoviens permet d'exprimer la vraisemblance conjointe des champs O et E sous la forme d'une distribution de Gibbs, [3] :

$$p(o,e) = \frac{1}{Z} e^{-U(o,e)} \tag{1}$$

où $U(o,e)$ est appelée fonction d'énergie et se décompose sous la forme :

$$U(o,e) = \sum_{c \in C} V_c(o,e) \tag{2}$$

C désigne l'ensemble des cliques c. Les cliques c sont des ensembles de points mutuellement voisins (au sens du système de voisinage ν choisi pour le modèle), et V_c est appelé potentiel d'interaction locale associé à la clique c. V_c présente la propriété remarquable de ne dépendre que des sites éléments de la clique c.

L'adoption du critère du Maximum A Posteriori (MAP) conduit à rechercher le champ des primitives $\hat{e}$ qui maximise la distribution jointe :

$$\hat{e} = arg \max_{e} p(o,e) \tag{3}$$

On voit, dans ce contexte de modélisation, que maximiser la distribution conjointe $p(o,e)$ est équivalent à minimiser la fonction d'énrgie $U(o,e)$ mêlant observations et primitives. Cette minimisation est un problème (difficile) d'optimisation globale qui peut se résoudre par des méthodes de relaxation stochastique (qui permettent théoriquement de converger vers le minimum global de U, [8]). Ces algorithmes sont toutefois coûteux et des solutions sous-optimales très satisfaisantes peuvent souvent être obtenues, à un coût moindre, grâce à des méthodes déterministes, [2, 6]. Des algorithmes de relaxation déterministe sont utilisés ici, avec de bons résultats expérimentaux.

2.3 Application de la modélisation markovienne à la détection du mouvement

L'algorithme de détection du mouvement considéré ici est décrit en détail dans [11]. Nous rappelons brièvement ici ses différentes composantes.

2.3.1 Définition des observations et des étiquettes

En détection du mouvement les étiquettes e à extraire sont des étiquettes binaires, qui décrivent localement l'état d'un point $s=(x,y)$:

$$e(s) = \begin{cases} 0 & \text{absence de mouvement au point s} \\ 1 & \text{mouvement au point s} \end{cases}$$

Deux familles différentes d'observations sont considérées:

- o_t sont les différences d'intensité entre deux images successives:

$$o_t(x,y) = |I(x,y,t) - I(x,y,t-dt)| = D(x,y) \tag{4}$$

- $\bar{o}_t$ est une carte binaire résultant de la détection de changements temporels par test d'hypothèse sur une fenêtre locale A, [10, 11]:

$$\bar{o}_t = \begin{cases} 0 & si \;\; R' < \lambda \\ 1 & si \;\; R' > \lambda \end{cases}$$

où

$$R' = \frac{1}{n^2}|\sum_{s\in A} D(s)^2| + \frac{1}{\sum_{s\in A}\Delta x^2}[(\sum_{s\in A}\Delta x D(s))^2 + (\sum_{s\in A}\Delta y D(s))^2]$$

Les coordonnées Δx et Δy sont définies par rapport au centre de la fenêtre A. Le seuil λ détermine la sensibilité du détecteur. S'il est trop faible il devient sensible au bruit (figure 1-a), s'il est trop grand la projection du masque de l'objet en mouvement ne sera pas complète (figure 1-b).

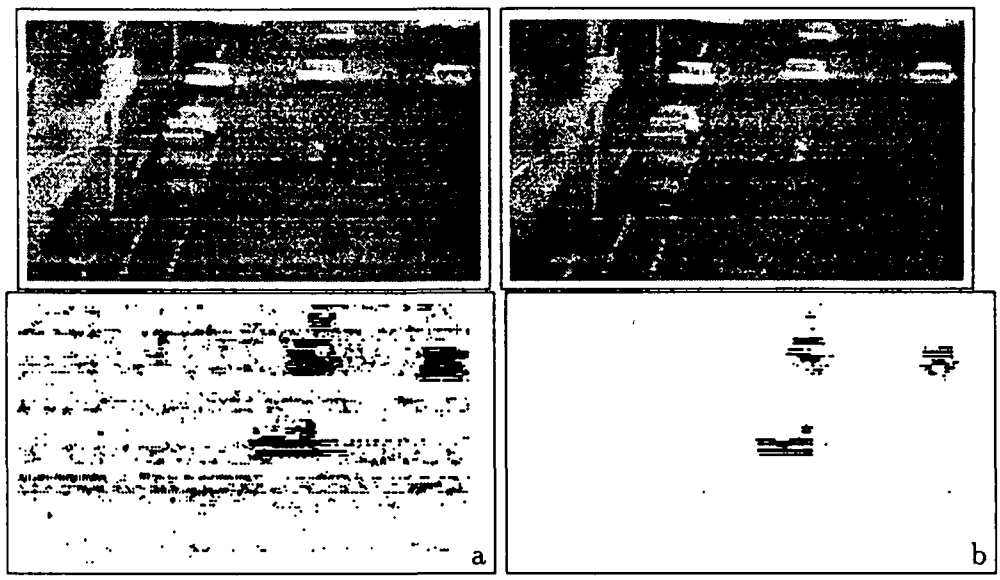


Figure 1: Détection des changements temporels

2.3.2 Critère de décision

Le critère de décision choisi est celui du maximum a posteriori (MAP). L'expression à maximiser s'écrit :

$$\max_{e_{t-dt},e_t} P(e_{t-dt}, e_t, o_t, \bar{o}_t) \tag{5}$$

où :

- e_t est l'étiquetage à l'instant t
- e_{t-dt} est l'étiquetage à l'instant *t-dt*
- o_t et $\bar{o}_t$ sont les observations décrites précédemment

et $P(e_{t-dt}, e_t, o_t, \bar{o}_t)$ désigne la distribution de probabilité conjointe, spécifiée sous la forme :

$$P(e_{t-dt}, e_t, o_t, \bar{o}_t) \sim exp(-U(e_{t-dt}, e_t, o_t, \bar{o}_t)) \tag{6}$$

On remarque qu'une décision conjointe sur les cartes d'étiquettes aux instants t et *t-dt* est adoptée pour permettre une bonne reconstruction du masque de l'objet à un instant donné.

2.3.3 Définition de la fonction d'énergie

La fonction d'énergie à construire dépend essentiellement du problème à traiter. Dans [11], elle est définie de la façon suivante :

$$U(e_{t-dt}, e_t, o_t, \bar{o}_t) = W_c(o_t, e_{t-dt}, e_t) + W_s(e_{t-dt}) + W_s(e_t) + W_\tau(e_{t-dt}, e_t, \bar{o}_t) \tag{7}$$

La fonction d'énergie se compose de quatre termes: un terme qui relie les observations avec l'étiquetage courant (W_ϵ), des termes d'énergie spatiale (W_s) et une énergie temporelle (W_τ). Ces différents termes expriment chacun une contrainte dans le problème de détection du mouvement. Ils se définissent comme suit :

1. *Energie* $W_\epsilon(o_t, e_{t-dt}, e_t)$.

   Elle exprime les relations statistiques entre observations o_t et les étiquetages aux instants $t - dt$ et t, suivant le modèle :

$$o_t(s) = \Psi(e_{t-dt}(s), e_t(s)) + n(s) \tag{8}$$

   où Ψ est une fonction qui modélise le changement temporel et $n(s)$ un bruit blanc gaussien centré de variance σ^2. La fonction Ψ prend trois valeurs possibles :

$$\Psi(e_{t-dt}(s), e_t(s)) = \begin{cases} 0 & si \;\; e_{t-dt}(s) = e_t(s) = 0 \\ m_1 & si \;\; e_{t-dt}(s) = e_t(s) = 1 \\ m_2 & si \;\; e_{t-dt}(s) \neq e_t(s) \end{cases} \tag{9}$$

   avec $0 < m_1 << m_2$

   $m_1 << m_2$ exprime que le glissement d'un objet sur lui-même (transition $e_{t-dt}(s) = e_t(s) = 1$) engendre une variation d'intensité plus faible que le passage d'un objet mobile vers le fond statique (transition $e_{t-dt}(s) \neq e_t(s)$).

   Sous des hypothèses gaussiennes sur le bruit $n(s)$, il vient :

$$W_\epsilon(o_t(s), e_{t-dt}(s), e_t(s)) = \frac{1}{2\sigma^2} \sum_{s \in S} [o_t(s) - \Psi(e_{t-dt}(s), e_t(s))]^2 \tag{10}$$

2. *Energie spatiale* $W_s(e_t)$

   L'énergie $W_s(e_t)$ exprime l'homogénéité spatiale des masques des objets mobiles. Elle permet d'éliminer les détections parasites liées au bruit.

   Comme l'estimation se fait simultanément sur les cartes d'étiquettes (à *t* et *t-dt*), $W_s(e_t)$ est définie sur ces deux cartes. L'énergie spatiale utilisée est associée à un modèle markovien classique, proposé par Geman, [8]. Le système de voisinage considéré est un 8-voisinage dont on retient uniquement les cliques à deux éléments (Fig. 2). Les potentiels associés à chaque couple de pixels qui appartient à une clique spatiale sont les suivants :

   - $V_{C_s} = \beta_s$ si $e_t(s_1) \neq e_t(s_2)$
   - $V_{C_s} = -\beta_s$ si $e_t(s_1) = e_t(s_2)$

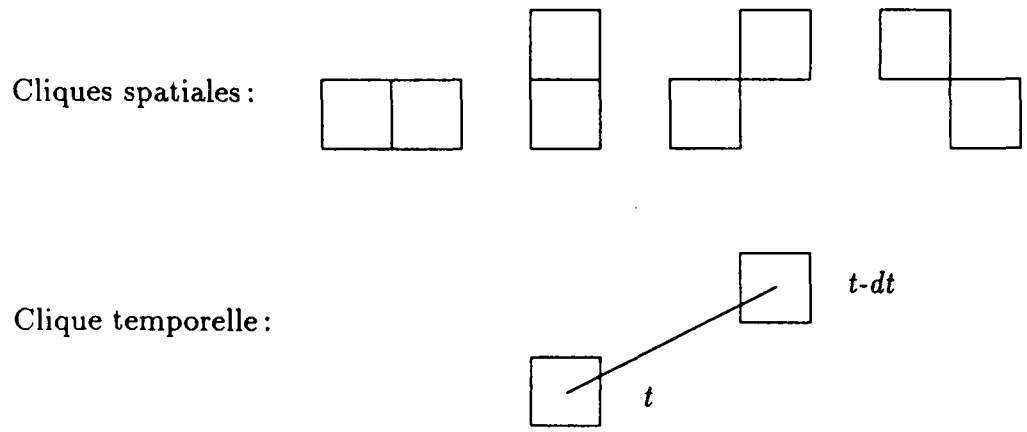


Figure 2: Systèmes de voisinages adoptés

avec β_s positif. L'énergie spatiale totale correspond à la somme des potentiels calculés pour l'ensemble des cliques spatiales de l'image(Fig. 2) :

$$W_s = \sum_{c_s \in C_s} V_{c_s} \tag{11}$$

3. *Energie temporelle* $W_\tau(e_{t-dt}, e_t, \bar{o}_t)$

   L'énergie temporelle porte sur les observations $\bar{o}_t$ obtenues par détection de changements temporels. Le seuil de ce détecteur est fixé à une faible valeur de façon à permettre la validation du glissement d'un objet sur lui-même. L'énergie temporelle contribue à la reconstruction du masque des objets mobiles à un instant donné, [11] :

$$W_\tau = \sum_{c_\tau \in C_\tau} V_\tau(e_{t-dt}(s), e_t(s), \bar{o}_t(s)) \tag{12}$$

   La fonction de potentiel V_τ est donnée en Table 1 avec $0 < \beta_\tau << \beta'_\tau$.

2.3.4 Méthodes de relaxation

L'algorithme permettant de minimiser la fonction d'énergie définie dans la section précédente (Equ. 7) est un algorithme itératif de relaxation.

Relaxation stochastique et déterministe

Les méthodes de relaxation déterministes partent généralement d'une configuration initiale et créent une nouvelle configuration qui diffère de la précédente en un seul site de l'image. On choisit la remise à jour locale qui maximise la descente en énergie. La remise à jour des sites de l'image est poursuivie jusqu'à convergence (vers un minimum local dans

$(e_{t-dt}, e_t, \bar{o}_t)$	$V_\tau(e_{t-dt}, e_t, \bar{o}_t)$
(0,0,0)	$-\beta_\tau$
(0,0,1)	$+\beta_\tau$
(1,0,0)	$+\beta'_\tau$
(1,0,1)	$+\beta'_\tau$
(0,1,0)	$+\beta_\tau$
(0,1,1)	$-\beta_\tau$
(1,1,0)	$+\beta_\tau$
(1,1,1)	$-\beta_\tau$

Table 1: Potentiel associé à l'énergie temporelle

ce cas, [2]). Les techniques de relaxation stochastique permettent, par contre, de converger théoriquement vers la solution optimale (correspondant au minimum global de la fonction d'énergie), [8]. Elles nécessitent toutefois un grand nombre d'itérations pour converger. Une technique déterministe a donc été retenue ici. Des résultats très satisfaisants avec ce type de relaxation peuvent être obtenus à condition d'initialiser judicieusement l'algorithme. Comme on peut le voir à la figure 3, si la configuration initiale est située dans la zone A, la méthode déterministe converge vers un minimum local. Par contre, si la configuration initiale est dans la zone B, on converge vers un minimum global.

Une bonne solution consiste, dans notre cas à initialiser le champ des étiquettes à un instant donné t par celui obtenu à l'instant précédent.

Choix du site à visiter: La pile d'instabilité et l'algorithme HCF

Différentes façons de remettre à jour les sites de l'image ont été proposées: un balayage séquentiel des sites, [8], un tirage aléatoire des sites, [8] ou la gestion d'une pile d'instabilité (algorithme HCF) [6].

Cette dernière méthode se concentre directement sur les points instables pour lesquels l'énergie n'est pas minimale. Elle converge donc très rapidement quand le nombre de points à traiter est petit par rapport à la taille de l'image. Les deux autres méthodes, par contre, traitent systématiquement tous les pixels de l'image.

Dans notre cas, comme peu de pixels sont modifiés entre deux cartes d'étiquettes successives, la méthode de la pile est la plus adéquate. Pour créer la pile, on calcule l'énergie locale pour tous les points et on vérifie si elle est minimale. Dans le cas contraire, on calcule un coefficient d'instabilité défini comme la différence entre l'énergie actuelle et l'énergie minimale. On ordonne tous les points selon les valeurs croissantes de ce coefficient. A chaque itération, on prend le point au sommet de la pile (le plus instable) et on modifie son étiquetage $e_{t-dt}(s)$ et $e_t(s)$ de façon à minimiser l'énergie locale. Comme on est dans un cadre markovien, cette

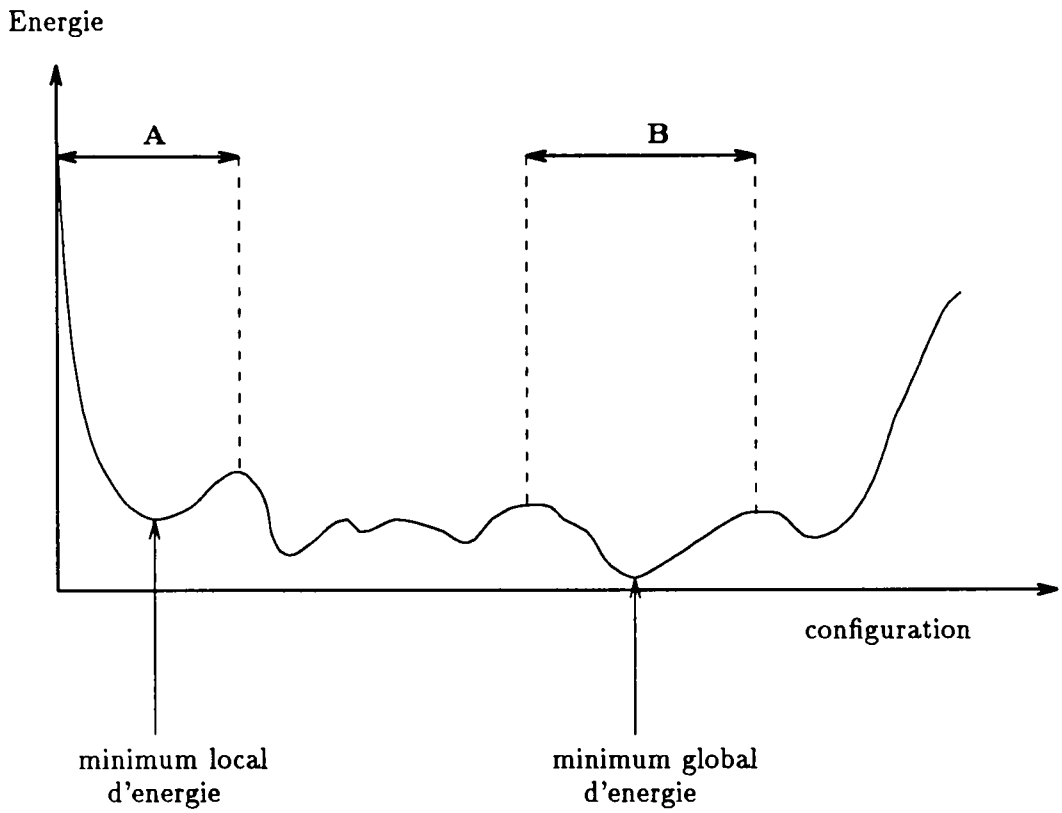


Figure 3: Fonction d'énergie et minima locaux

modification d'étiquetage amène seulement à un changement de l'énergie des huit voisins. Ce changement d'étiquette peut donc modifier l'instabilité des voisins, qui sont intégrés dans la pile. Ce procédé est répété jusqu'à ce que la pile soit vide. Remarquons que pour une convergence rapide, il est important que le nombre de points instables soit faible, sinon la gestion de la pile devient trop coûteuse.

2.4 Synoptique de l'algorithme

Pour conclure cette partie concernant la présentation de l'algorithme nous présentons le synoptique (Fig. 4) correspondant à son implémentation séquentielle.

L'algorithme vient tout d'abord lire en mémoire les deux premières images de la séquence $I(x,y,t)$ et $I(x,y,t-dt)$, puis calcule les observations correspondantes :

- $o_t(x,y) = |I(x,y,t) - I(x,y,t-dt)|$
- $\bar{o}_t(x,y)$

$$\bar{o}_t = \begin{cases} 0 & si \quad R' < \lambda \\ 1 & si \quad R' > \lambda \end{cases}$$

où

$$R' = \frac{1}{n^2} | \sum_{s \in N} D(s)^2 | + \frac{1}{\sum_{s \in N} \Delta x^2} [(\sum_{s \in N} \Delta x D(s))^2 + (\sum_{s \in N} \Delta y D(s))^2]$$

L'étape suivante consiste à calculer l'énergie pour chaque pixel ainsi que l'énergie minimale correspondante. Le coefficient d'instabilité est la différence entre ces deux valeurs. A partir de ces coefficients on crée la pile en ordonnant les points selon leur instabilité (les plus instables au sommet). On entre alors dans la phase de gestion de la pile : on change les étiquettes (e_{t-dt} et e_t) pour le point au sommet de la pile, puis on recalcule la nouvelle valeur de l'énergie de ses voisins. Si ceux-ci deviennent instables on les rajoute à la pile. On recommence le même procédé pour les points suivants de la pile jusqu'à ce qu'elle soit vide.

La dernière étape consiste à écrire la carte finale des étiquettes e_{t-dt} dans un fichier (on répète cette procédure pour toutes les images de la séquence).

2.5 Résultats obtenus avec l'algorithme séquentiel

Nous montrons ici les résultats de la version séquentielle obtenus sur deux séquences différentes. Chaque image est composée de deux trames de taille 128 × 256. Nous avons traité uniquement les trames impaires. Ces séquences ont été acquises avec une caméra grand public et ensuite numérisées. Elles sont donc bruitées ce qui permet de tester la robustesse de l'algorithme. Les valeurs des paramètres que nous avons utilisées (pour les deux séquences) sont : $m_1 = 4$, $m_2 = 15$, $\beta_s = 5$, $\beta_\tau = 30$, $\beta_{\tau'} = 100$. Le temps de traitement moyen pour l'algorithme séquentiel sur une station de travail SUN (sparc I) est de 30 secondes pour une image 512 × 512. Un traitement en temps réel (vidéo) nécessite la mise en œuvre de cet algorithme sur une architecture parallèle que nous étudions dans le paragraphe suivant.

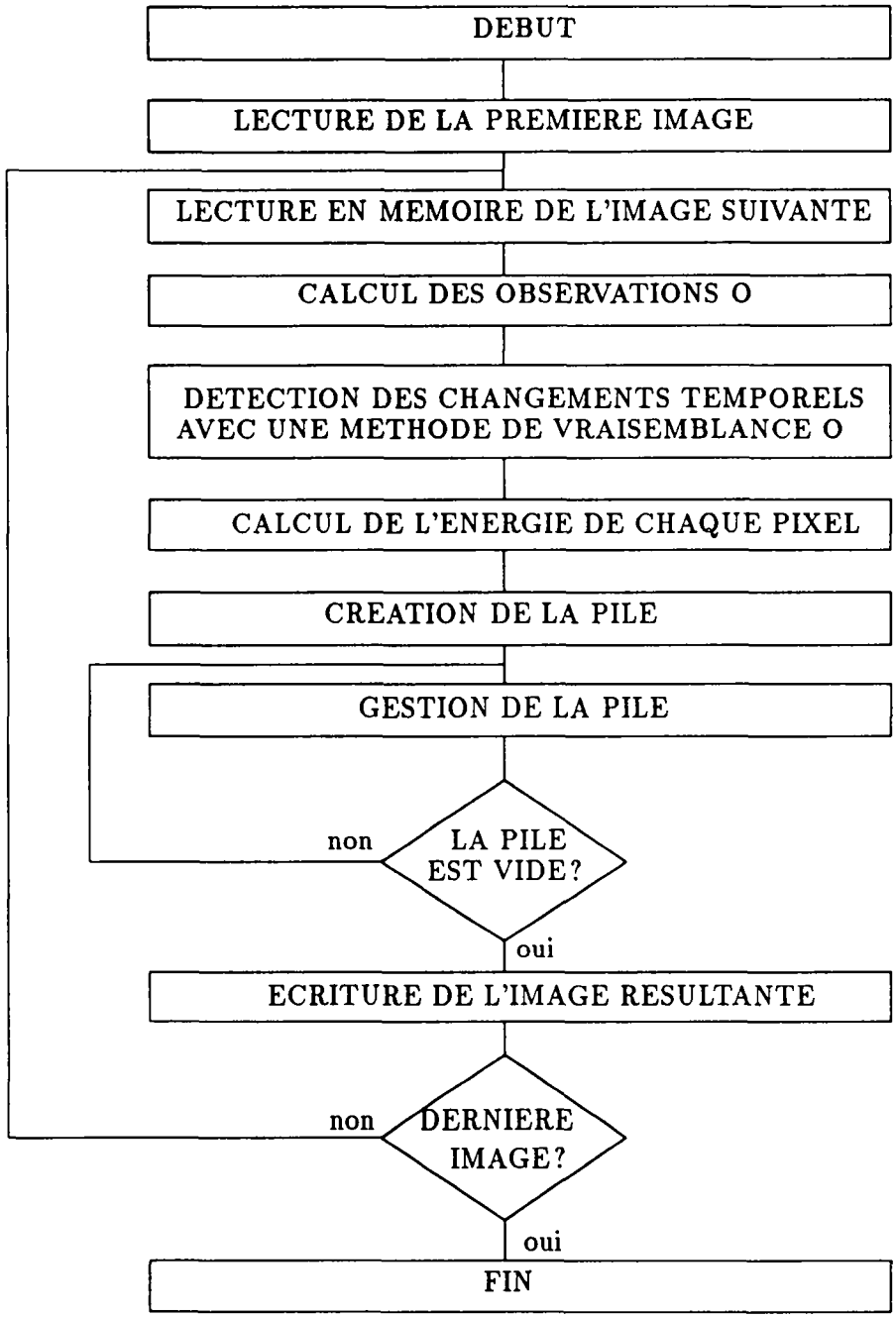


Figure 4: Synoptique de l'algorithme

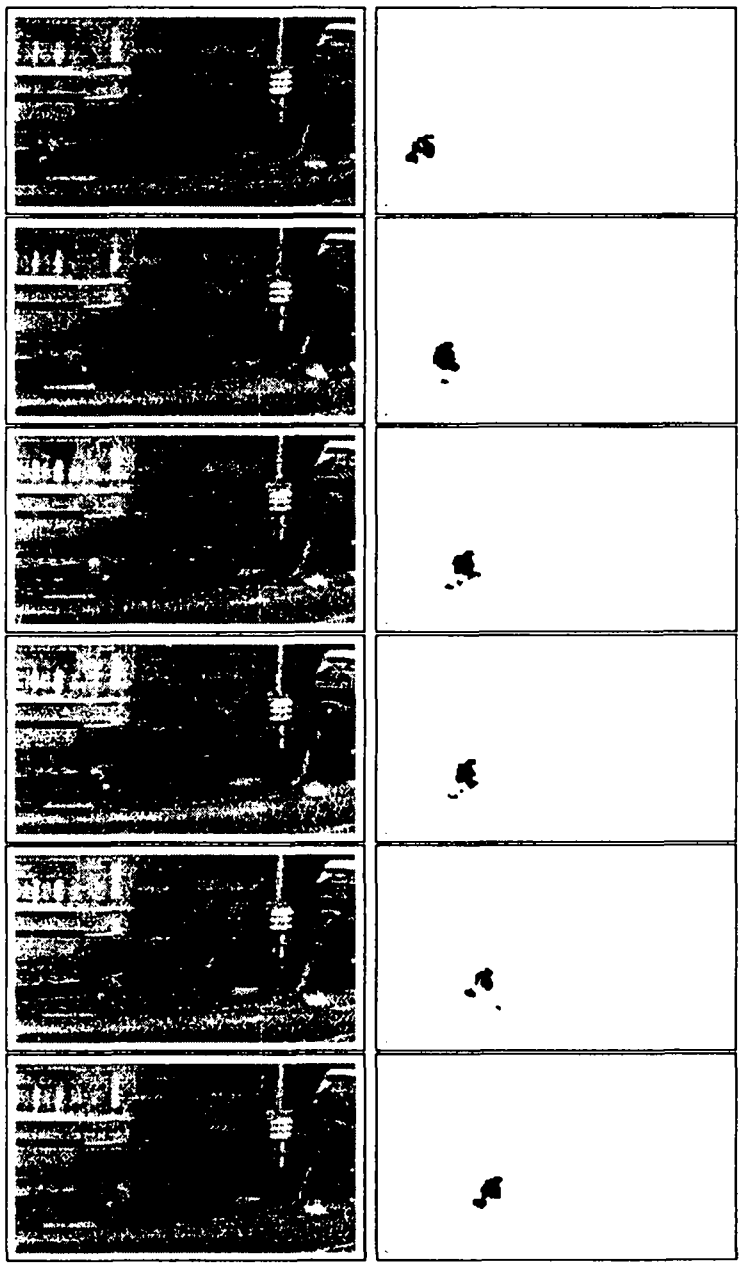


Figure 5: Séquence "piéton"

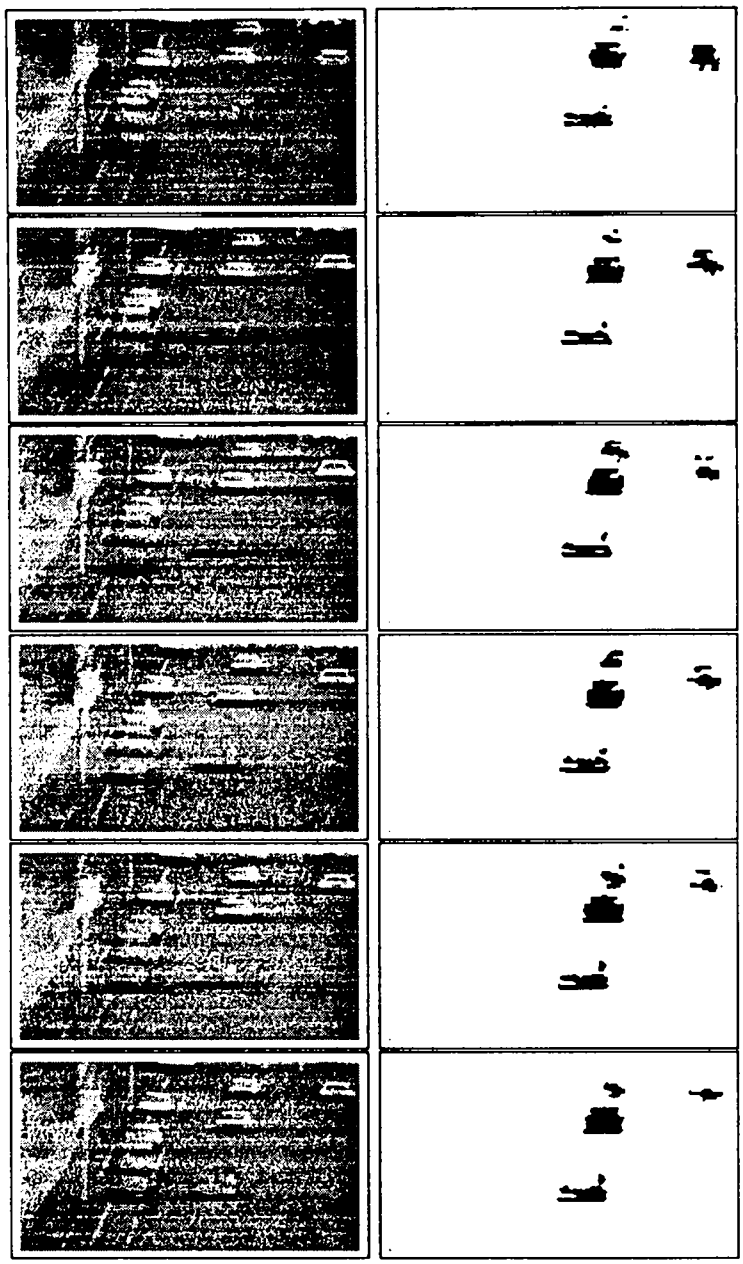


Figure 6: Séquence "voitures"

3 Parallélisation de l'algorithme

Nous nous intéressons dans ce paragraphe à la conception d'une version parallèle de l'algorithme, décrit dans le paragraphe 2, pour des architectures parallèles MIMD à mémoire distribuée. Après une brève introduction aux techniques de parallélisation, nous présentons plusieurs stratégies pour partitionner les données et distribuer les calculs sur les processeurs de l'architecture afin d'obtenir un algorithme efficace pour ce type d'architecture.

3.1 Techniques de parallélisation

Parmi les différentes techniques de parallélisation, celle dite de "parallélisation par distribution des données" semble être la plus adaptée aux architectures parallèles à mémoire distribuée. Dans cette approche, les données exploitées par l'algorithme sont distribuées dans les mémoires locales des processeurs. Chaque processeur exécute les calculs qui concernent les données dont il est le propriétaire. Le concepteur doit déterminer une fonction de placement des données qui doit satisfaire les contraintes suivantes :

- **Equilibrer les charges** : pour obtenir de bonnes performances, les processeurs doivent travailler "utilement" pendant toute la durée d'exécution de l'algorithme.
- **Minimiser les communications** : l'échange de messages entre processeurs est une opération coûteuse en temps. On doit donc, dans la mesure du possible, réduire le nombre des communications. Il est préférable d'utiliser des messages de taille importante plutôt que plusieurs petits messages afin d'amortir le temps de latence associé à l'envoi d'un message. Cette latence est due à la traversée des couches du système d'exploitation lors de l'envoi d'un message ainsi qu'à l'initialisation des composants matériels utilisés pour le transfert des messages.
- **Eviter l'interblocage** : les communications provoquent très souvent des problèmes d'interblocage. Cela se produit par exemple quand deux processeurs distincts attendent chacun un message de l'autre. Il se produit aussi quand il ne reste plus de place dans la mémoire locale des processeurs pour allouer les tampons nécessaires à la gestion des messages.

Pour satisfaire ces contraintes, le placement des données peut être réalisé statiquement ou dynamiquement. Si les accès aux données sont connus a priori, une partition statique est réalisable. Sinon, la répartition devra se faire pendant l'exécution de l'algorithme. Elle évoluera au cours de l'exécution afin d'équilibrer au mieux les charges.

3.2 Choisir une stratégie

Le schéma d'implantation parallèle que nous avons choisi pour paralléliser l'algorithme de détection de mouvement est celui dirigé par les données. L'image est distribuée sur les différentes mémoires locales de manière à avoir un bon équilibrage des charges. Nous devons

	Séq. "piéton"		Séq. "voitures"	
nb images	15	30	15	30
observations	31,5%	32,3%	25,5%	26,5%
énergie initiale	29,2%	31,1%	24,8%	25,3%
gestion de la pile	19,5%	15,3%	32,5%	30,9%
total	80,2%	78,7%	82,8%	82,7%

Table 2: Profil d'exécution du programme séquentiel

aussi essayer de minimiser les communications inter-processeurs. La première étape à réaliser est l'estimation des temps d'exécution de chaque partie de l'algorithme pour dégager le partitionnement adéquat pour les modules les plus lents.

3.2.1 Temps d'exécution des modules de l'algorithme

Afin d'estimer le temps d'exécution de chaque module de l'algorithme, nous avons réalisé des profils d'exécution. A partir de ces profils, nous avons constaté qu'il y a trois modules qui prennent, au total, plus de 75% du temps d'exécution du programme. Le reste est réparti entre la lecture et l'écriture des images, la création de la pile et l'initialisation des variables pour chaque image de la séquence. Le tableau 2 montre les pourcentages de temps de ces trois modules sur deux types de séquences : la séquence "piéton" avec peu de mouvement et la séquence "voitures" avec davantage de zones en mouvement.

D'après le profil nous pouvons tirer plusieurs conclusions :

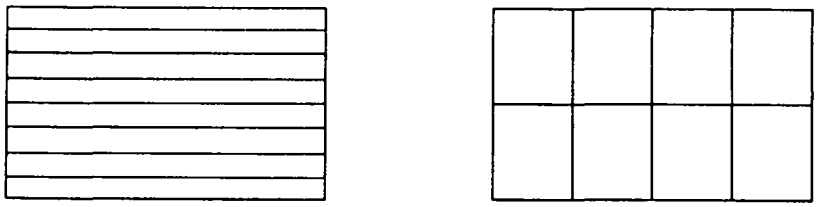
1. Les temps de calcul des observations et de l'énergie initiale dépendent seulement de la taille des images et non du type de séquence à traiter. Par contre, le temps de gestion de la pile dépend du nombre de zones en mouvement. Comme nous voulons paralléliser l'algorithme quelle que soit la séquence à traiter, nous devons en tenir compte.

2. De même, le pourcentage du temps consacré à la gestion de la pile décroît lorsque le nombre d'images de la séquence augmente. Cela est dû au fait que, pour les premières images, la configuration initiale des étiquettes est très éloignée de la configuration optimale (le nombre de points instables est donc important). Par contre, pour les images suivantes, le nombre de points instables se réduit aux frontières des objets mobiles.

3. La gestion de la pile est moins coûteuse que le calcul des observations ou de l'énergie initiale.

Nous considérons donc deux types de partitionnement :

- Partitionnement uniforme de l'image.



Bandes Rectangles

Figure 7: Partitionnement de l'image en bandes et rectangles

Cette solution réalise un bon équilibrage pour le calcul des observations et celui de l'énergie initiale. En effet, ces deux modules réalisent le même nombre de calculs sur tous les pixels de l'image. Chaque processeur doit donc travailler sur un nombre égal de pixels.

- Partitionnement concentré sur les zones instables.

  Pendant la gestion de la pile, les calculs se concentrent sur les zones en mouvement, alors qu'il n'y a aucun calcul sur la reste de l'image. Les processeurs peuvent alors se répartir les zones instables.

Le fait de privilégier un partitionnement plutôt qu'un autre entraîne une perte d'efficacité, ce qui nous amène à implanter les deux partitionnements pendant le traitement de chaque image. Nous commençons donc par partitionner l'image en sous-images de même taille pour le calcul des observations et de l'énergie initiale ; puis, nous réalisons un nouveau partitionnement pour la gestion de la pile. Pour l'image suivante, on recommence avec un partitionnement uniforme.

3.2.2 Phase de calcul des observations et de l'énergie initiale

Les calculs des observations et de l'énergie initiale s'effectue sur tous les pixels de chaque image de la séquence. Nous proposons donc un partitionnement statique uniforme (toutes les sous-images sont de même taille). Nous allons aussi utiliser ce partitionnement pour les modules de lecture des images et d'écriture des résultats (la carte des étiquettes). Deux solutions peuvent être envisagées pour le partitionnement uniforme :

- division en bandes,
- division en rectangles.

La division en bandes horizontales paraît plus simple car le balayage des images est horizontal. La lecture et l'écriture des sous-images se fera de façon continue. De plus, chaque processeur possède seulement deux voisins alors que la division en rectangles nécessite quatre

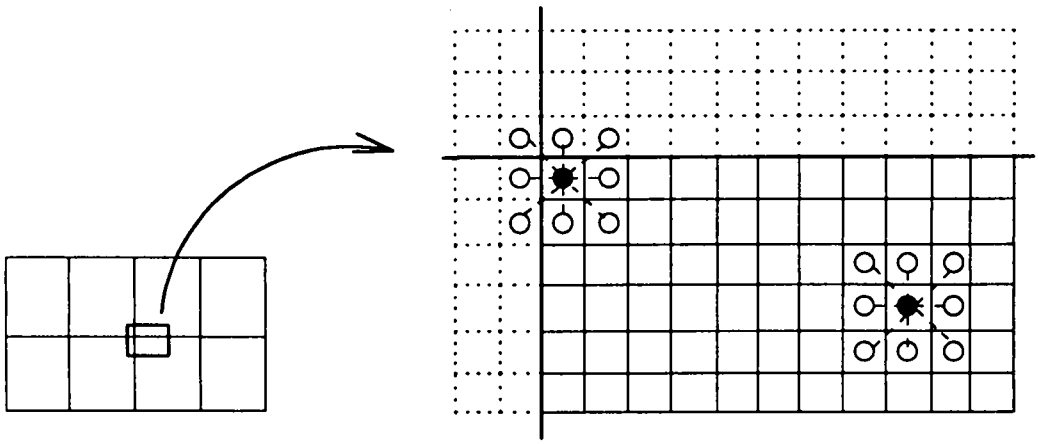


Figure 8: Problème des frontières

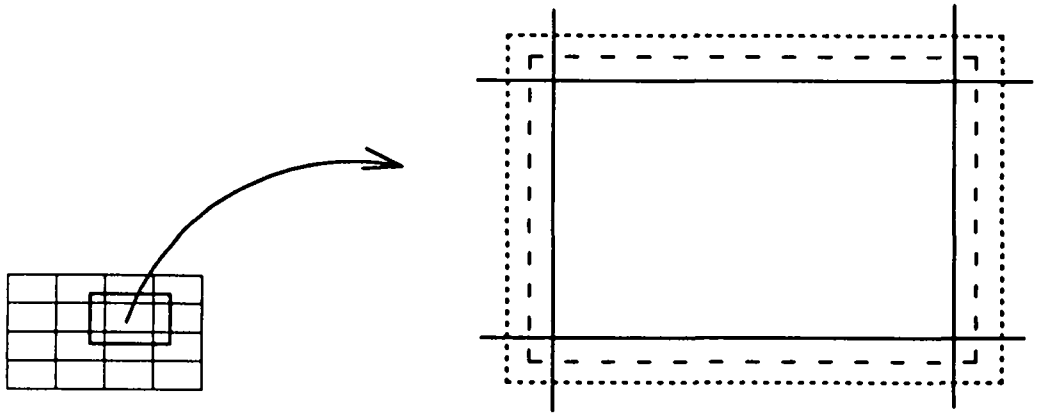
voisins par processeur. Une division en bandes horizontales permet donc de réduire les communications par rapport à une division en rectangles. Ces deux stratégies ont été implémentées. Nous avons rejeté la division en bandes verticales puisqu'elle est plus complexe que celle en bandes horizontales. Remarquons également que, pendant le calcul des observations et de l'énergie, chaque processeur doit connaître les caractéristiques des voisins de tous les pixels sur lesquels il doit faire les calculs. La figure 8 montre qu'avec le partitionnement en cours, un processeur ne peut pas effectuer les calculs sur les points situés à la frontière de sa sous-image puisqu'il ne connaît pas les valeurs des voisins.

Pour résoudre ce problème nous pouvons envisager de réaliser des communications entre processeurs voisins pour échanger les pixels situés sur la frontière si nécessaire. Une meilleure solution consiste à utiliser une stratégie de recouvrement : chaque processeur va acquérir quelques lignes supplémentaires correspondant aux frontières des nœuds voisins de manière à supprimer les communications. Pour cela, il suffit de prendre deux lignes de recouvrement à chaque frontière pour éviter toute communication. En effet, dans le calcul de l'énergie nous avons besoin de connaître la valeur des observations aux pixels extérieurs de la frontière. Pour calculer les observations correspondantes à ces derniers nous avons besoin d'une deuxième ligne de recouvrement (figure 9).

Les cartes des observations et les coefficients d'instabilité que nous avons extraits de cette phase sont identiques à ceux du programme séquentiel. L'équilibre de charges est ainsi obtenu.

3.2.3 Phase de gestion de la pile

Afin de réaliser un partitionnement de l'image entre les différents processeurs, nous devons estimer les zones où le volume des calculs est important. Les zones instables changent pour chaque image de la séquence, par conséquent le partitionnement devra être dynamique. Dans la phase de calcul de l'énergie, nous comptons le nombre de points instables dans chaque



—— Sous-image assignée à un nœud. Calcul de l'énergie.
- - Première ligne de recouvrement. Calcul des observations.
····· Partie effective de l'image stockée dans la mémoire d'un noeud.

Figure 9: Recouvrement nécessaire pour éviter les communications

nœud. Ce nombre correspond en fait aux zones initialement instables. Il n'est donc pas identique au nombre final des points traités par la pile. En effet, quand l'étiquetage d'un pixel change, ses voisins peuvent devenir instables et donc peuvent être rajoutés à la pile. Inversement, des points instables peuvent devenir stables. Cependant cette information est suffisante pour avoir une connaissance générale des zones les plus instables et réaliser ainsi un bon équilibrage des charges. Nous assignerons donc aux processeurs qui ne présentent pas de points instables initialement, des parties des régions instables traitées par les autres processeurs, ceci afin d'assurer l'équilibrage des charges dans la gestion de la pile. Chaque nœud recevra un message indiquant le nombre de processeurs qui vont travailler sur sa partie d'image, puis il va la diviser en bandes non uniformes afin d'équilibrer les charges. Cette nouvelle division en bandes ne dépend pas du choix de partitionnement de la phase précédente (en bandes ou rectangles). La figure 10 montre un exemple dans lequel le processeur numéro 4 divise sa sous-image en trois parties, une pour lui-même et les autres pour les processeurs 2 et 7 respectivement.

L'étape suivante consiste à envoyer aux processeurs "esclaves" toutes les données relatives à la gestion de la pile dans les bandes d'image correspondantes. Ceci implique l'envoi de la valeur des étiquettes (e_{t-dt} et e_t), des observations et de l'énergie de tous les pixels de la bande. Les processeurs qui ne présentent pas de zones instables attendront le message contenant les données de la région qu'ils devront traiter. Chaque processeur va alors créer sa propre pile et la gérer indépendamment des autres. Lorsqu'un processeur vide la pile d'une zone dont il n'est pas propriétaire, il envoie le résultat final au processeur propriétaire de celle-ci. Par ailleurs, on retrouve ici le problème des frontières vu dans la phase précédente.

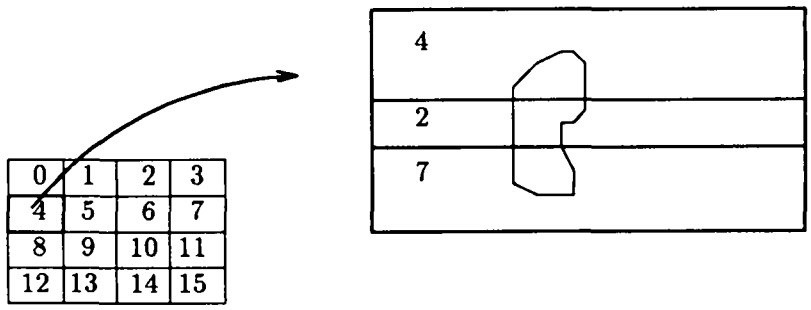


Figure 10: Division en trois parties de la sous-image du processeur 4

Quand un processeur change l'étiquette d'un pixel de la frontière, l'énergie de quelques pixels du nœud voisin change et ces derniers peuvent devenir instables. Cette fois-ci, comme il s'agit d'un algorithme de relaxation, nous ne pouvons pas obtenir le même résultat que celui obtenu avec le programme séquentiel sans faire appel aux communications. Le recouvrement peut seulement nous aider à corriger quelques erreurs. Nous avons testé deux parallélisations différentes, la première utilisant des communications pour résoudre les problèmes aux frontières et l'autre (sous-optimale) utilisant seulement un recouvrement.

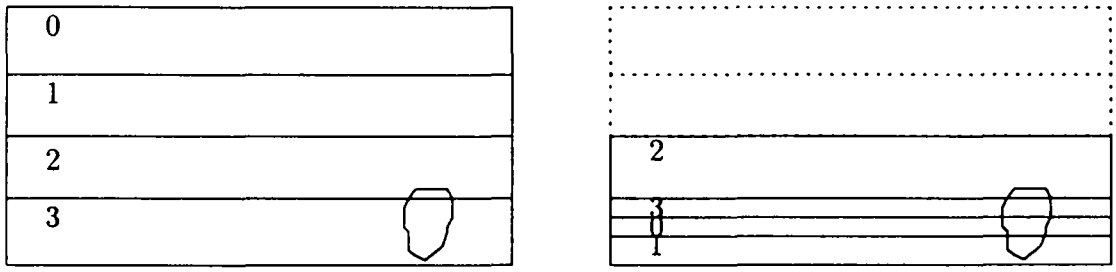
3.3 Solutions proposées

3.3.1 Partitionnement en bandes avec communications

La division en bandes de la première phase de calcul des observations paraît la solution la plus adaptée si on utilise des communications dans la phase suivante de gestion de la pile. De cette façon, chaque processeur présente seulement deux voisins. Par conséquent le contrôle de ces communications sera plus facile qu'avec une division en rectangles. Le but recherché dans cette parallélisation est d'arriver au même résultat d'étiquetage que le programme séquentiel. Nous pouvons prévoir d'avance que l'efficacité ne sera pas optimale dans ce cas. La première phase des observations se déroule donc en utilisant un partitionnement uniforme en bandes utilisant le recouvrement de deux lignes de pixels.

En ce qui concerne la gestion de la pile, il y a une nouvelle division en bandes des sous-images qui présentent des zones instables. Chaque processeur va aussi enregistrer sur sa mémoire locale une ligne supplémentaire de la frontière. Ceci permet la lecture de l'énergie des points de la frontière sans utiliser de communications. Nous devons rappeler que la lecture de ces énergies est nécessaire pour pouvoir changer les étiquettes des points situés à la frontière. Cependant les pixels de la ligne supplémentaire ne seront pas ajoutés à la pile. Ils seront en effet traités par le processeur voisin.

Le mécanisme de communication envisagé est le suivant. Quand un processeur modifie l'étiquetage d'un pixel de la frontière, l'énergie des huit voisins change (figure 12). Ce processeur peut seulement changer l'énergie des cinq voisins qui lui appartiennent. Il envoie donc au processeur voisin la nouvelle valeur de l'étiquetage modifié pour qu'il remette à jour l'énergie



Partitionnement des observations Partitionnement de la pile

Figure 11: Partitionnement en bandes

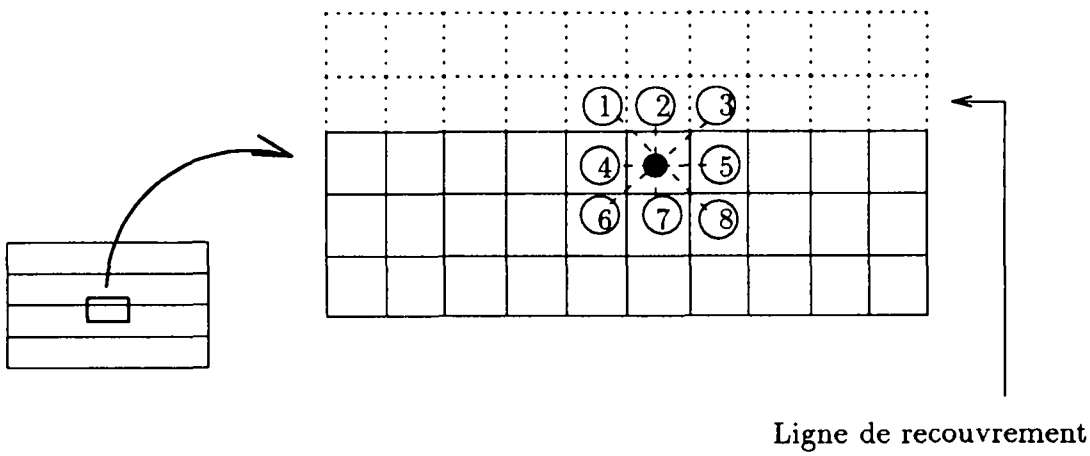


Figure 12: Communications à la frontière de deux nœuds

des trois pixels qui restent (pixels 1, 2 et 3, figure 12). A la réception de ces valeurs, celui-ci va recalculer l'énergie pour ces trois pixels, puis les remettre dans sa pile si ces derniers deviennent instables. Enfin, il va envoyer la nouvelle valeur de l'énergie de ces trois pixels au premier processeur. Cela permet un nouveau changement d'étiquette. Les communications que nous avons utilisées dans cette phase sont non bloquantes: les processeurs n'interrompent la gestion de leur pile que sur la réception d'un message d'un nœud voisin. La méthode de relaxation utilisée dans cet algorithme est déterministe. Par conséquent, le fait de diviser la pile peut mener à un minimum d'énergie local différent de celui auquel on arrive avec la version séquentielle. Cela entraîne une configuration finale des étiquettes différente de celle du programme séquentiel mais visuellement très proche.

3.3.2 Partitionnement en rectangles sans communications

Dans la section précédente, nous avons vu que l'utilisation des communications pour gérer la pile ralentit beaucoup le programme. Nous avons donc essayé de gérer la pile sans communications. Cependant, dans ce cas, le résultat final que nous obtenons pour l'étiquetage diffère de celui du programme séquentiel. Nous allons donc chercher un compromis entre la vitesse d'exécution et les erreurs dans l'étiquetage. Comme il n'y a pas de communications, nous pouvons utiliser la division en rectangles. Ce partitionnement nous permet d'avoir un meilleur équilibrage des charges. Ainsi, la phase de calcul des observations va utiliser un partitionnement uniforme en rectangles, puis à la phase suivante ces rectangles vont être divisés en bandes. La figure 13 montre un exemple de ces partitionnements dans le cas de huit processeurs.

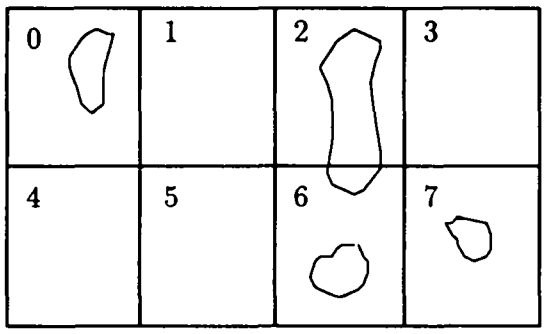
4 Expérimentations

4.1 Machine d'expérimentation: l'iPSC/2

L'iPSC/2 est une architecture parallèle à mémoire distribuée de type MIMD. Il est constitué d'un ensemble de processeurs de calcul connectés selon une topologie hypercube

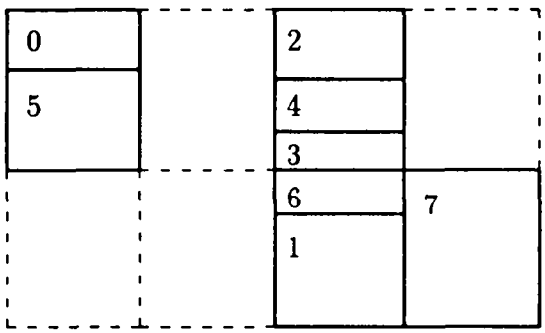
4.1.1 Architecture

Chaque nœud de l'hypercube est composée d'un processeur Intel 80386 à 16 Mhz permettant de traiter les instructions à la vitesse de 4 MIPS. Un coprocesseur 80387 est utilisé pour les opérations flottantes et offre une performance de 0.3 Mflops. La mémoire disponible est de 4 Mo et peut être étendue jusqu'à 16 Mo. Les communications inter-processeurs sont gérées par un coprocesseur spécialisé (DCM) [12] assurant les fonctions de transfert et de routage des messages. Chaque coprocesseur DCM dispose de 8 liens. Sept sont utilisés pour la connexion avec les processeurs voisins (la configuration maximale est d=7). Le huitième lien est utilisé pour connecter processeurs d'entrée/sortie gérant par exemple des unités de disque. Ce dernier lien sert, dans le cas du nœud 0, à la connexion avec l'ordinateur frontal. L'ordinateur frontal est un PC AT386 fonctionnant sous Unix System V ce qui permet la



Distribution uniforme des processeurs

• Zones instables



Nouvelle distribution non uniforme des processeurs

Figure 13: Partitionnement en rectangles

connexion de plusieurs utilisateurs sur l'iPSC/2. Il permet la compilation des programmes aussi que leur chargement dans les nœuds.

4.1.2 Environnement logiciel

Le système d'exploitation de l'iPSC/2 est composé de deux parties : la première réside dans le processeur frontal et permet à plusieurs utilisateurs de se connecter sur un sous-ensemble des nœuds. Il existe une librairie UNIX avec de nouvelles commandes qui facilitent la gestion de l'hypercube. Elles permettent, par exemple, l'allocation d'un sous-ensemble des nœuds aux différents utilisateurs ou le chargement, l'exécution et l'arrêt des processus dans les nœuds de l'hypercube. La deuxième partie correspond au système d'exploitation qui s'exécute sur chaque nœud de l'hypercube, appelé NX/2. Elle se charge de plusieurs tâches :

- Gestion des processus permettant un fonctionnement multiprocessus sur chaque nœud de l'hypercube.
- Gestion des communications : envoi et réception de messages entre processeurs en mode bloquant ou non bloquant.
- Gestion de la mémoire. Elle permet une allocation dynamique de la mémoire pour les processus et aussi pour les tampons alloués au système pour la gestion des communications.

4.1.3 Les communications entre processeurs

Le système d'exploitation NX/2 offre deux types de communication : bloquant ou non bloquant. En mode de communication bloquant, le processus qui exécute une demande de réception de message s'arrête et attend jusqu'à ce que le message soit arrivé. Le processus qui envoie un message en mode bloquant reprend son activité une fois que le message est parti vers le destinataire. Il n'attend donc pas que le message soit arrivé à son destinataire. En mode non bloquant, le processus n'est pas bloqué lorsque celui-ci exécute une demande de réception ou d'envoi de message. Cela permet de faire dans certains cas des communications et des calculs en parallèle. Toutes les communications sont gérées par le DCM (Direct-Connect Routing Module) indépendamment des processeurs de chaque nœud. En ce qui concerne le routage des messages, celui-ci a été implanté par une technique de commutation de circuits de type "cut-through". Cette technique établit dans un premier temps un chemin entre deux processeurs puis, dans un deuxième temps le message est transmis au destinataire. C'est une technique similaire au système téléphonique. Elle offre une latence plus faible que les commutations par paquets.

4.2 Métriques pour l'évaluation des performances

Les performances des solutions que nous avons proposées peuvent être évaluées en utilisant deux critères : l'accélération et l'efficacité. L'accélération représente le gain réel en vitesse obtenu avec l'algorithme parallèle par rapport à une exécution séquentielle. Elle est définie

nb. nœuds	*accélération*	*efficacité*
2	1,65	0,825
4	2,9	0,725
8	5	0,625
16	7,5	0,469

Table 3: Performance de la solution avec communications

par :

$$S_p = \frac{T_s}{T_p} \tag{13}$$

où

- T_s= Temps d'exécution de l'algorithme séquentiel
- T_p= Temps d'exécution de l'algorithme parallèle sur p processeurs.

L'efficacité nous donne la valeur relative du gain sans dépendance avec le nombre de processeurs utilisé.

$$E = \frac{S_p}{p} \tag{14}$$

L'accélération maximale que nous pouvons obtenir est $S_p = p$. Par conséquent $E < 1$.

4.3 Partitionnement en bandes avec communications

La table 3 présente les performances obtenues pour la séquence "voiture" avec cette méthode. Pour calculer les performances, nous n'avons pas pris en compte le temps d'exécution des modules de lecture et d'écriture des images puisque la configuration actuelle de l'iPSC/2 à l'IRISA ne permet pas l'implantation parallèle des procédures d'entrée/sortie. Tous ces procédures se déroulent donc séquentiellement à partir du frontal.

4.4 Partitionnement en rectangles sans communication

Nous avons testé l'algorithme en nous servant d'un nombre variable de lignes de recouvrement. Nous avons conclu que la meilleure solution est celle qui utilise une seule ligne de recouvrement. En effet, cette solution permet d'obtenir une efficacité optimale tout ayant un nombre d'erreurs dans l'étiquetage peu nombreux. Nous pouvons expliquer ce petit nombre d'erreurs par la forme de la fonction d'énergie :

$$U(e_{t-dt}, e_t, o_t, \bar{o}_t) = W_\epsilon(o_t, e_{t-dt}, e_t) + W_s(e_{t-dt}) + W_s(e_t) + W_\tau(e_{t-dt}, e_t, \bar{o}_t) \tag{15}$$

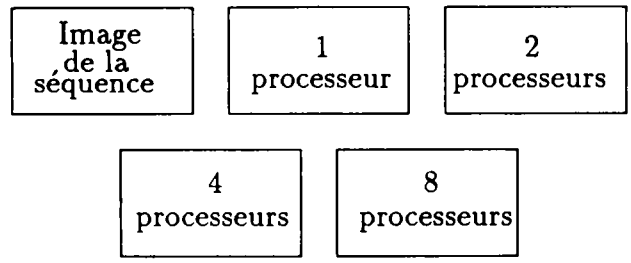


Figure 14: Plan des résultats de la séquence "piéton"

où :

- W_ϵ est l'énergie qui relie les observations avec l'étiquetage
- W_s est l'énergie spatiale
- W_τ est l'énergie temporelle

Le fait de ne pas utiliser de communications affecte seulement l'énergie spatiale. De plus, l'énergie spatiale correspondant à un pixel dépend de ses huit voisins et nous avons seulement des erreurs pour trois voisins. L'étiquetage est donc satisfaisant même en l'absence de communications dans la phase de gestion de la pile. Nous devons aussi remarquer que l'étiquetage obtenu par le programme séquentiel ne correspond pas exactement à la projection complète de l'objet mobile. Le nombre d'erreurs introduites par la version parallèle est alors insignifiant.

4.5 Résultats

Nous présentons dans les paragraphes suivants, les résultats obtenus pour les séquences "piéton" et " voitures" en utilisant une stratégie de partitionnement en rectangles sans communication. La taille des images est de 256 × 128 pixels. Les séquences ont été testées avec un nombre variable de processeurs.

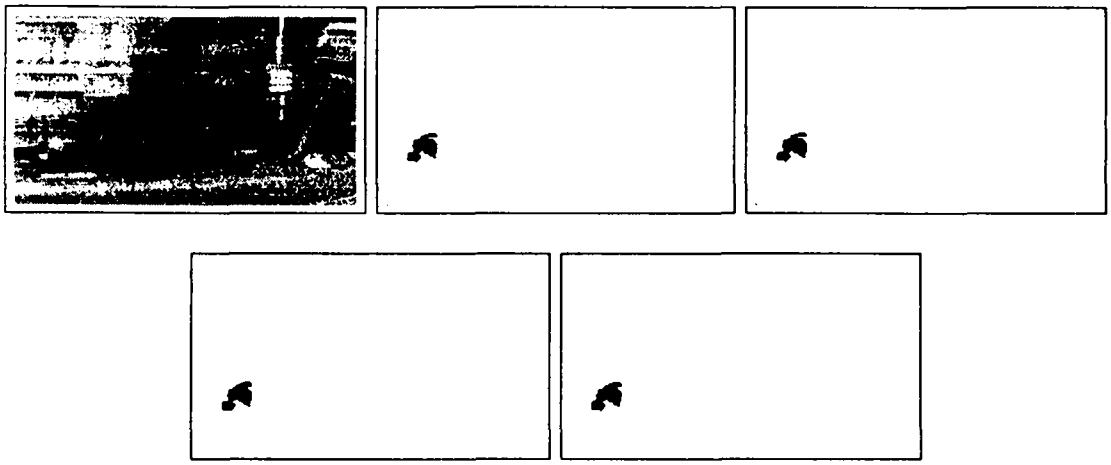
4.5.1 Séquence "piéton"

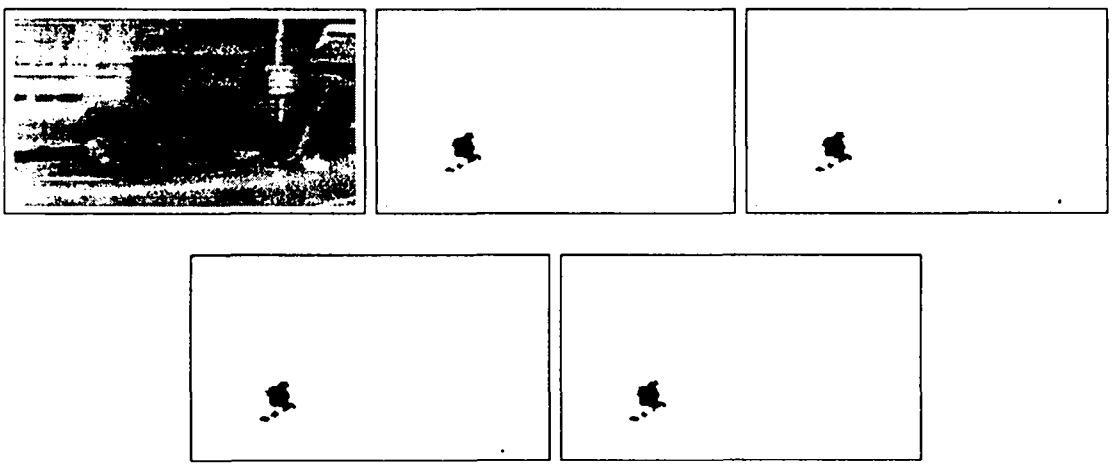
Pour cette séquence, 8 processeurs ont été utilisés. La zone en mouvement est trop petite pour utiliser un nombre de processeurs plus important avec de bonnes performances.

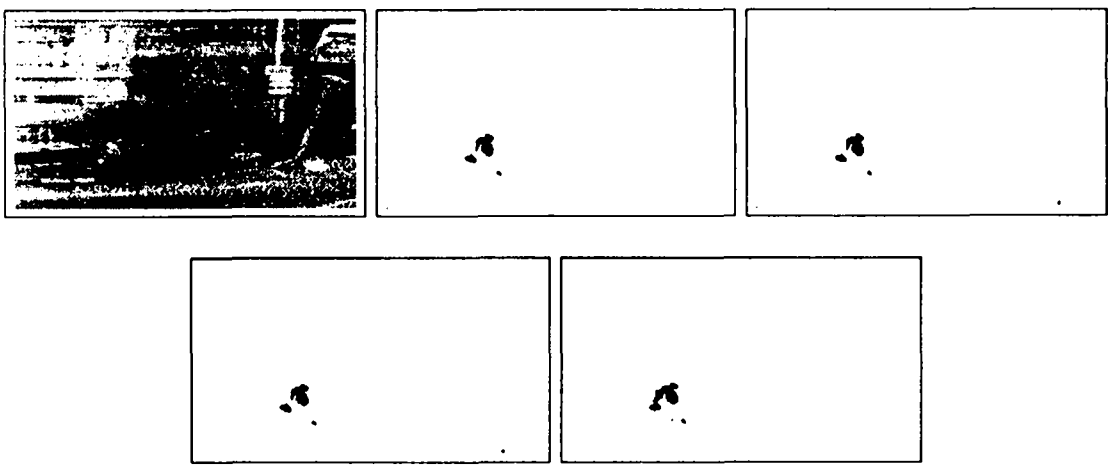
4.5.2 Séquence "voiture"

Cette séquence présente plus de zones en mouvement que la précédente ce qui nous a permis de la tester avec 16 processeurs.

A partir des résultats, on constate que la différence entre les résultats des versions séquentielle et parallèle est pratiquement imperceptible. La qualité diminue lorsque le nombre de







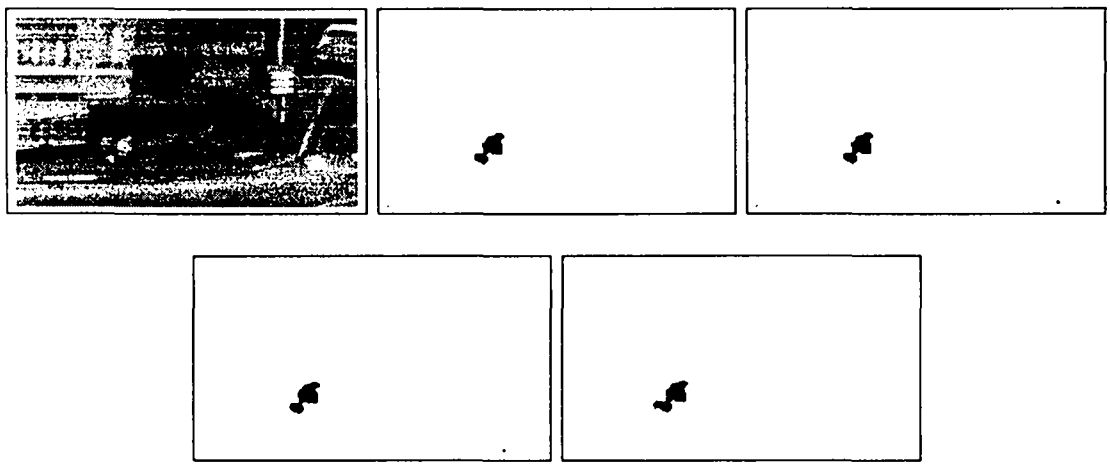


Figure 15: Séquence "piéton"

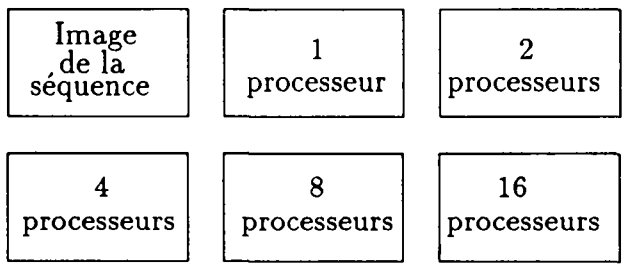


Figure 16: Images des résultats de la séquence "voiture"

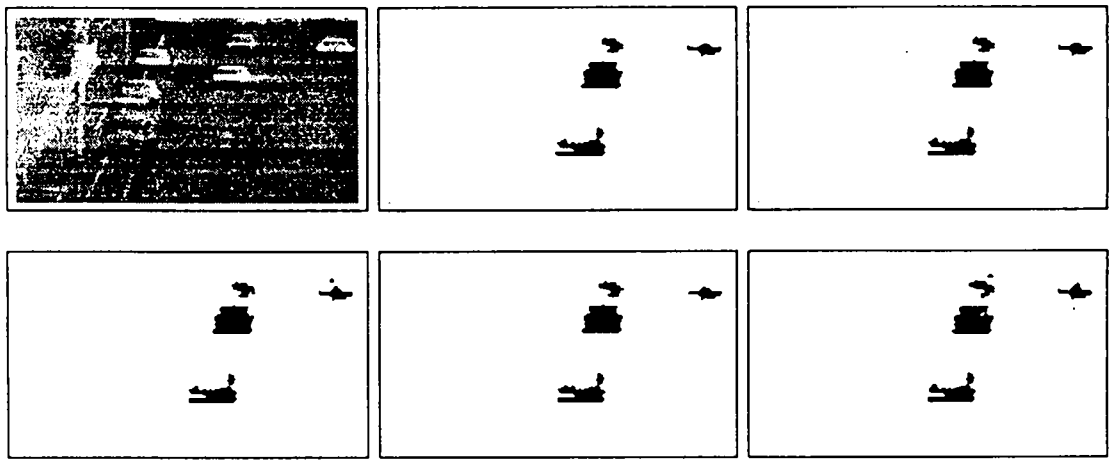


Figure 17: Séquence voiture

nb. nœuds	*accélération*	*efficacité*
2	1,9	0,95
4	3,4	0,85
8	6,1	0,76
16	8,9	0,55

Table 4: Performance de la solution sans communication

processeurs augmente. Ceci est dû à l'augmentation du nombre de frontières et à la diminution de la taille des zones instables à gérer par chaque processeur. Le nombre de processeurs, pouvant être utilisé, est donc limité. Cette limite est fonction de la taille des images de la séquence et du nombre de zones en mouvement.

4.6 Performances

La table 4 présente l'accélération et l'efficacité obtenues avec la version parallèle sans communication. On peut observer que l'efficacité obtenue est bonne. Cependant la taille limitée des images traitées entraîne une forte diminution de l'efficacité quand le nombre de processeurs augmente. Il faut utiliser des images de taille plus importante pour avoir des résultats satisfaisants avec un grand nombre de processeurs. La figure 18 présente les courbes d'accélération pour les deux versions parallèles envisagées. On constate l'amélioration obtenue avec la version choisie qui n'utilise pas de communications pendant la phase de gestion de la pile.

Nous constatons aussi que l'efficacité diminue avec le nombre de processeurs. Ceci est dû à la partie de gestion du partitionnement de la pile. En effet, cette partie prend le même temps indépendamment du nombre de processeurs et devient donc prépondérante quand ce nombre est grand (figure 19).

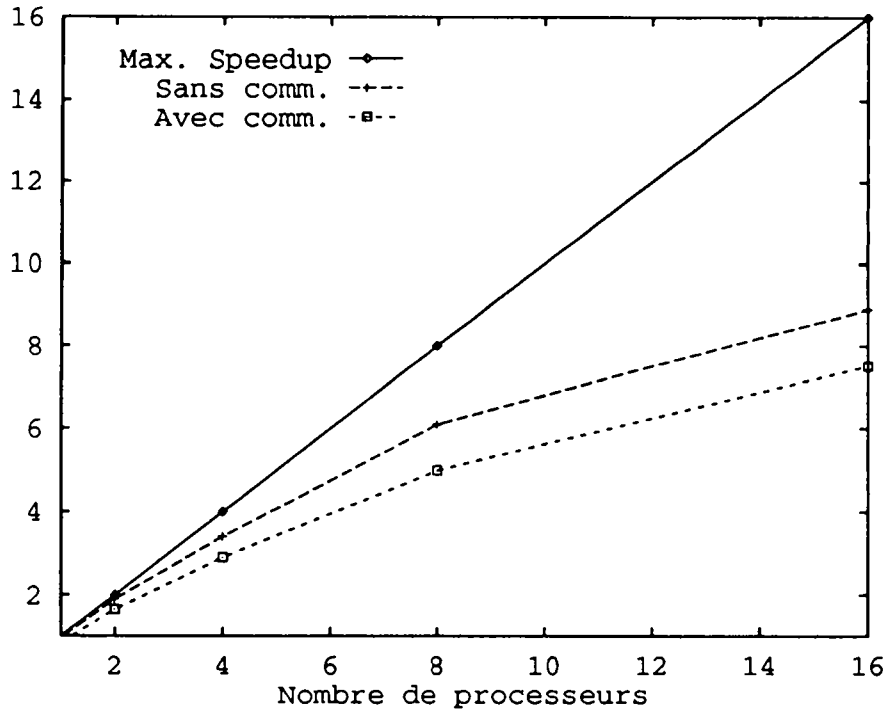


Figure 18: Accélération de l'algorithme parallèle

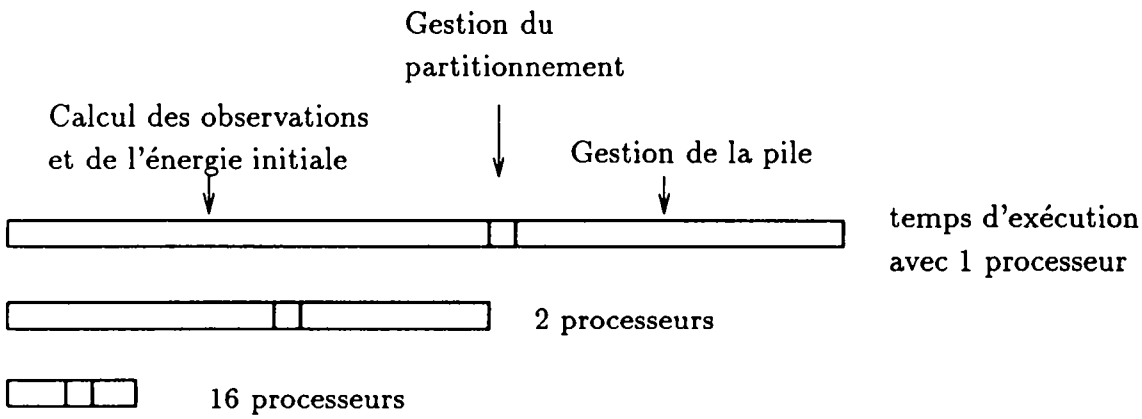
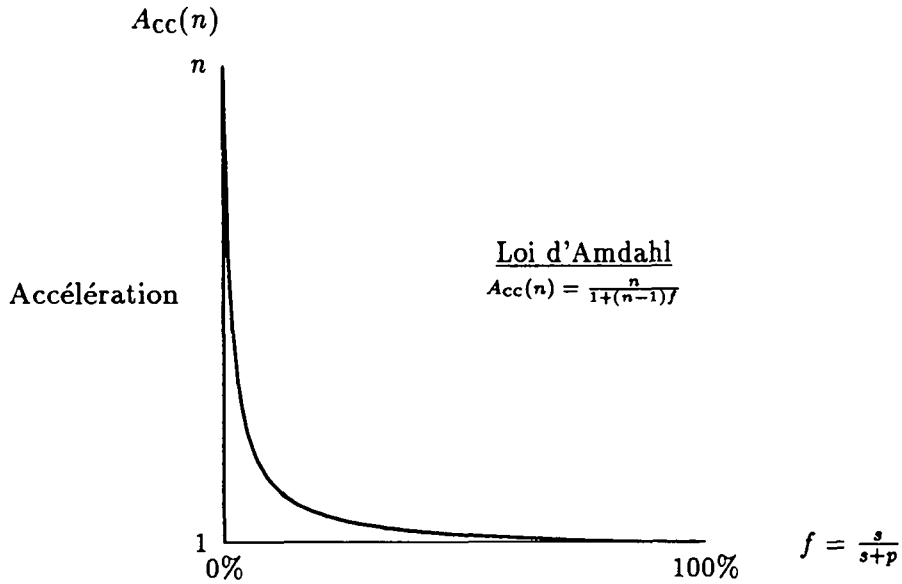


Figure 19: Section critique de l'algorithme parallèle



n: nombre de processeurs

s: temps d'exécution de la partie séquentielle de l'algorithme

p: temps d'exécution de la partie parallèle de l'algorithme

f: fraction séquentielle de l'algorithme

Figure 20: Loi d'Amdhal

Ceci est conforme à la loi d'Amdahl : il n'est pas souhaitable d'envisager des architectures massivement parallèles lorsque la fraction séquentielle d'un algorithme est importante. En effet, celle-ci devient alors prépondérante dans le temps d'exécution parallèle (figure 20).

5 Conclusion et perspectives

Dans ce rapport, une méthode de parallélisation générale pour les algorithmes HCF d'analyse d'images basés sur une modélisation markovienne a été décrite.

Les performances obtenues avec la version parallèle de l'algorithme markovien prouve que ce type d'algorithme, reposant sur la gestion d'une pile d'instabilité, est bien adapté à une implantation sur une architecture MIMD à mémoire distribuée. La dégradation des performances, lorsque le partitionnement des données est géré sans communication entre les processeurs, s'est avéré négligeable pour le modèle particulier considéré ici. Le développement en cours des architectures MIMD permettra aux prochaines générations de machines des performances encore accrues. Nous envisageons en particulier la parallélisation d'autres algorithmes basés sur des modèles markoviens, plus coûteux en temps de calcul, [9].

Des recherches actuellement en cours, concernant la mise en œuvre d'un dispositif de mémoire virtuelle partagée sur une architecture à mémoire distribuée (iPSC/2) ouvre par

ailleurs de nouvelles perspectives quant à la parallélisation des algorithmes de cette famille.

References

[1] J.K. AGGARWAL and N. NANDHAKUMAR. On the computation of motion from sequences of images - a review. *Proc. IEEE*, Vol. 76, No 8: pages 917–935, 1988.

[2] J. BESAG. On the statistical analysis of dirty pictures. *J. Royal Statist. Soc.*, Vol. 48, Serie B, No 3: pages 259–302, 1986.

[3] J. BESAG. Spatial interaction and the statistical analysis of lattice systems. *J. Royal Statist. Soc.*, Serie B, Vol. 36: pages 192–236, 1974.

[4] P. BOUTHEMY. Modèles et méthodes pour l'analyse du mouvement dans une séquence d'images. *Technique et Science Informatiques*, Vol. 7, No 6: pages 527–546, 1988.

[5] P. BOUTHEMY and P. LALANDE. Detection and tracking of moving objects based on a statistical regularization method in space and time. In *Proc. First European Conference on Computer Vision*, pages 307–311, Springer, Antibes, France, April 1990.

[6] P.B. CHOU and C.M. BROWN. The theory and practice of bayesian image modeling. *Int. J. Comp. Vis.*, Vol 4: pages 185–210, 1990.

[7] E. FRANCOIS and P. BOUTHEMY. Multiframe-based identification of mobile components of a scene with a moving camera. In *IEEE Int. Conf. Computer Vision Pattern Recognition*, pages 166–172, Hawaii, June 3-6 1991.

[8] S. GEMAN and D. GEMAN. Stochastic relaxation, Gibbs distributions and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 6, No 6: pages 721–741, November 1984.

[9] F. HEITZ and P. BOUTHEMY. Multimodal motion estimation and segmentation using Markov random fields. In *Proc. 10th Int. Conf. Pattern Recognition*, pages 378–383, Atlantic City, June 1990.

[10] HSU, Y. Z., NAGEL, H.-H, and REKERS, G. New likelihood test methods for change detection in image sequences. *Comput. Vision, Graphics, Image Processing*, Vol. 26: pages 73–106, 1984.

[11] P. LALANDE. *Détection du mouvement dans une séquence d'images selon une approche markovienne; application à la robotique sous-marine.* PhD thesis, Université de Rennes I, Mars 1990.

[12] NUGENT S.F. The iPSC/2 Direct-Connect Communications Technology. *Third Conference on Hypercube Concurrent Computers and Applications. Pasadena, California*, 51–60, Janvier 1988.

LISTE DES DERNIERES PUBLICATIONS INTERNES PARUES A L'IRISA

PI 661 REACHABILITY ANALYSIS ON DISTRIBUTED EXECUTIONS
Claire DIEHL, Claude JARD, Jean-Xavier RAMPON
Juin 1992, 18 pages.

PI 662 RECONSTRUCTION 3D DE PRIMITIVES GEOMETRIQUES PAR VISION ACTIVE
Samia BOUKIR, François CHAUMETTE
Juin 1992, 40 pages.

PI 663 FILTRES SEMANTIQUES EN CALCUL PROPOSITIONNEL
Raymond ROLLAND
Juin 1992, 22 pages.

PI 664 REGION-BASED TRACKING IN AN IMAGE SEQUENCE
François MEYER, Patrick BOUTHEMY
Juin 1992, 50 pages.

PI 665 CORRECTNESS OF AUTOMATED DISTRIBUTION OF SEQUENTIAL PROGRAMS
Cyrille BARREAU, Benoît CAILLAUD, Claude JARD, René THORAVAL
Juin 1992, 32 pages.

PI 666 AGREGATION FAIBLE DES PROCESSUS DE MARKOV ABSORBANTS
James LEDOUX, Gerardo RUBINO, Bruno SERICOLA
Juillet 1992, 30 pages.

PI 667 MODELES D'EVALUATION DE LA FIABILITE DU LOGICIEL ET TECHNIQUES DE VALIDATION DE SYSTEMES DE PREDICTION : ETUDE BIBLIOGRAPHIQUE
James LEDOUX
Juillet 1992, 76 pages.

PI 668 TWO COMPLEMENTARY NOTES ON SKEWED-ASSOCIATIVE CACHES
André SEZNEC
Juillet 1992, 10 pages.

PI 669 PARALLELISATION D'UN ALGORITHME DE DETECTION DE MOUVEMENT SUR UNE ARCHITECTURE MIMD
Fabrice HEITZ, Sergui JUFRESA, Etienne MEMIN, Thierry PRIOL
Juillet 1992, 34 pages.

ISSN 0249 - 6399